Case Studies in Moving Multi-Threaded Workstation Applications from RISC/UNIX to the Intel Architecture

June 1998

Order Number: 283035-001

TABLE OF CONTENTS

LIST OF TABLES

DISCLAIMERS AND RESTRICTIONS

Performance report notice

THIS TEST REPORT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Information in this document is provided in connection with Intel products. No license, express or implied, by Intel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Pentium II Processors may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

The hardware manufacturer remains solely responsible for the design, sale and functionality of its product, including any liability arising from product infringement or product warranty.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference:

www.intel.com/procs/perf/limits.htm or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

SPECint95 and SPECfp95 benchmark tests reflect the performance of the microprocessor, memory architecture and compiler of a computer system on compute-intensive, 32-bit applications. SPEC benchmark tests results for Intel microprocessors are determined using particular, well-configured systems. These results may or may not reflect the relative performance of Intel microprocessor in systems with different hardware or software designs or configurations (including compilers). Buyers should consult other sources of information, including system benchmarks, to evaluate the performance of systems they are considering purchasing. For more information about SPEC95, including a description of the systems used to obtain these test result, and other information about microprocessor and system performance and benchmarks, visit Intel's World Wide Web site at www.intel.com or call 1-800-628-8686.

ABSTRACT

Analyst reports indicate that sales of workstations based on the Intel Architecture now exceed sales of traditional RISC/UNIX workstations. More and more of these Intel workstations are being purchased with multiple processors, opening up opportunities for Enterprise users and Independent Software Vendors (ISVs) to deliver higher performing, multi-threaded applications. This means application developers must give new consideration to the costs and benefits of developing parallelism in their application software. In this paper, we look at several ISV applications and the performance scaling benefits that come from threading applications to take advantage of parallelism on Intel Architecture multiprocessor workstations. We address part of the overall subject by discussing the cost-effective use of shared memory parallelism development tools and procedures, specifically on Intel Architecture workstations, to produce hardware independent software applications that are parallelized for use on multiprocessor workstations and servers.

INTRODUCTION

The main benefit of using computers is in reducing the overall time to solve problems or to take advantage of an opportunity. There are, of course, costs associated with taking full advantage of computers so there is great interest from end users and ISVs in increasing the **Total Benefits of Ownership (TBO)/Total Cost of Ownership (TCO)** ratio.

For many Enterprise users, the **TBO** has come to mean **performance** in terms of total time to design/solution; the components of this time are human time and system time, and these depend on software effectiveness. **TCO** breaks into **initial hardware and software costs**, and **support and maintenance**. In many Enterprises, software effectiveness is now being dealt with competitively by outsourcing ISV application software. In Enterprises, **initial software costs** for high performance computing are related to their suppliers' (ISVs') development costs, including **portability** over the platforms they use (UNIX* and Windows* NT), **time to market** in **parallelizing** applications, **performance tuning**, **debugging** and **Q/A**.

ISVs are guided by a benefit/cost ratio similar to that of Enterprise users: **Total Benefit of Development (TBD)/Total Cost of Development (TCD)**, and TCD contributes to the initial software costs (see above) of Enterprise users. ISVs are always looking for ways to get a competitive edge, including performance improvements made possible by delivering applications on symmetric multiprocessing workstations and servers. As with Enterprise developers, ISV development and maintenance costs must be managed over multiple platforms. More than Enterprise users, ISVs face costs associated with maintaining applications over time on systems from multiple hardware vendors. ISVs must bear these TCDs in mind as they price products, and this feeds directly into the TBO/TCO considerations of Enterprise users when evaluating the acquisition of software.

More applications are becoming available on Intel Architecture workstations as ISVs understand the market opportunity and take advantage of cost-effective development tools. Below is a sampling of opportunity classifications that ISVs are addressing as they develop applications that take advantage of parallelism on Intel Architecture workstations:

- Supercomputer applications of a few years ago now run reasonably well on workstation systems so the range of applicability of Intel Architecture systems is growing.
- New markets develop for supercomputer applications once they are available on Intel Architecture workstation systems, increasing the sales volume substantially.
- PC applications of the past can find new markets when made available on much faster Intel Architecture workstation systems, also increasing sales volume.

- RISC/UNIX desktop applications may find much larger markets when ported to broadly available, high performance Intel Architecture workstation systems.

This paper discusses these subjects in the context of the **performance benefit** and **development costs** of parallel applications for multiprocessor Intel Architecture workstations. The scalability results were obtained from Intel Architecture workstations configured with two Pentium® II Xeon™ processors with 512 KB cache, and 512 MB main memory. The recently announced Intel Pentium II Xeon Processor is Intel's most powerful line of processors to date and brings new performance to mid-range and higher-end workstations and departmental and enterprise servers.

All test cases were done using realistic data sets in cooperation with the developers of the applications and the average speedup across these Intel Architecture/Windows NT systems is reported in the tables below for each application.

PARALLEL SOFTWARE ENGINEERING

Two methods of shared memory parallel (SMP) programming have been widely used in the past. The low-level approach is to "hand thread" an application using the parallel control mechanisms provided by POSIX threads or the Win32 Threading API. This requires programmers to implement the parallel control structure of a program by placing calls to the thread library in their application. Often, the low-level nature of such programming has forced ISVs to invent higher-level control routines that call the thread library. This simplifies the parallel programming process but isolates the ISV with a unique programming system.

The other approach to parallel programming has been based on the use of parallelism directives added to languages in the form of comment statements. Although directives were unified by the ANSI X3H5 proposal a decade ago, the standard was not completed so various companies implemented directives that diverged from the X3H5 proposal. In October 1997, several companies announced support for the OpenMP* standard which offers important programming innovations for directive based application development. (Intel is a sponsor of OpenMP.) Most important among the innovations is the orphaning of directives which allows directives to be placed at the highest level of a program structure, thereby implementing a parallel decomposition that crosses subroutine boundaries. The power of OpenMP has convinced many ISVs to adopt it, including those who had written their own threads-wrapping subroutine libraries in the past. OpenMP is being broadly adopted as a programming model for development of parallel application software; for more information, please see the OpenMP web page (www.openmp.org).

Beyond a powerful programming model, tools are essential to reduce **TCD**, but there has been a traditional lack of effective tools for parallel programming. KAI's (Kuck and Associates, Inc., web page: www.kai.com) KAP/Pro Toolset* for Windows NT for OpenMP was used in all of the work discussed below. This Toolset contains Guide, which processes OpenMP directives for NT or UNIX systems, GuideView, which provides a detailed profile of the components of parallel performance for each part of the program, and Assure, which verifies the correctness of a parallel program relative to the serial version of the program. It also contains KAP for autoparallelizing those parts of an application that do not require manual attention, and directive translators for moving OEM-specific directives into OpenMP. The use of these tools will be illustrated in dealing with the **TCD** and **TBO** aspects of several applications.

The KAI tools, and certain standard application development practices, were used on the example applications in the following way. This paper focuses on items 2, 3, and 4 below but it is nevertheless useful to state that adding parallelism to applications is part of a larger process:

1. Each effort started with stable software, and software engineers who were familiar with the source code, compiler requirements and utility requirements of each application
2. Each application enabled parallelism using directives
3. As needed, debugging was done on the parallel code
4. As needed, performance tuning was done
5. As needed, quality assurance procedures were applied

The first point may seem obvious but it is worth stressing. In parallelizing the applications presented in this paper, developers started with stable code that they understood. In an actual development setting, new features (and bug fixes) would be developed from time to time. It is not the subject of this paper, but innovation and repair can continue – while parallelization efforts proceed – by making sure that new features conform to the programming model of the application and by following good build and code-stream management practices.

Again, it is not the focus of this paper but compiler, datatype, and system utility issues need to be understood as part of the parallelization effort. Concerning compilers, some applications may take advantage of certain extensions or require inter-language communications. Regarding datatypes decisions for floating-point and integer-representation need to be made. Finally, applications being brought to the Intel Architecture may use system utilities such as memory allocation operations, timing routines, byte addressable I/O and numerical/math libraries. High quality compilers, libraries and utilities are readily available from third parties for use on Intel Architecture workstations running Windows NT.

As mentioned at the beginning of this section, there are two primary approaches to enabling parallelism. One is to 'hand code' it and the other is to use directives. In some of the examples below, parallelism was already built into the application using directives. These were updated using the OpenMP directives as supported in the use of the KAI tools. In these cases, the update was done rather quickly (a few days to a few weeks) by using KAI's Guide.

Debugging parallel programs is frequently difficult. Using print statements or multiple instances of standard debuggers can be cumbersome. While multi-threaded debuggers are making their way to Windows NT, debugging in the examples in this paper was assisted with the use of KAI's Assure tool found in the KAP/Pro Toolset. Application developers built their programs with the Assure instrumentor (looks like a normal compile) which instruments the serial program to emulate an ideal parallel processor. When applications are run with typical data sets, instrumented programs will produce a database of suspicious events that occurred on the ideal processor Assure is emulating.[1] Once these events are identified into the database, programmers can go through them to take appropriate corrective action.

After debugging, the parallelized applications in this paper were tuned for performance. In tuning parallel applications, one important thing to note is that performance gains depend directly on the fraction of the application that can be run in parallel. One useful way to estimate this is with Amdahl's law: [1]

$$\text{Performance Improvement} = \frac{T}{f * \left(\frac{T}{P}\right) + (1 - f) * T}$$

Where: T = Scalar execution time

f = Percentage of parallel code in the application

P = Number of processors

For example, if 75% of application code can be run in parallel and the scalar execution of the application code is 60 minutes, the potential performance improvement on a dual-processor Intel Architecture workstation would be 1.6 or a speed-up of 60% over the non-parallelized application.

With this understanding, software engineers can measure the actual performance and begin tuning. In the examples in this paper, KAI's parallel performance analysis tool called GuideView was used to analyze performance derived from the use of OpenMP directives. GuideView provides information on system performance bottlenecks such as:

- Synchronization time in locks and critical sections
- Loop imbalance due to the scheduling algorithm chosen
- Sequential time in code sections remaining as parallelization candidates

When bottlenecks were identified, opportunities for performance improvement were addressed by modifying the directives and/or modifying the application source code.

EXAMPLES

Below are three demonstrations of applying the KAP/Pro Toolset to technical workstation software development for parallel Intel/NT systems. Again, these results were gathered by running the application software on uni-processor and dual-processor workstations based on the Intel Pentium® II Xeon™ processor (400 MHz).

Fluent Inc., FLUENT*

Fluent Inc. (www.fluent.com) is a leading ISV developing computational fluid dynamics (CFD) software that is used worldwide for a broad range of engineering analyses by major industrial companies. FLUENT, one of their key applications, has run successfully on UNIX-based SMP systems for several years and has been available on single processor Intel NT systems for two years. Fluent Inc.'s cost of supporting parallelism on multiple platforms is managed by KAI's KAP/Pro Toolset, which provides **portability** between RISC/UNIX and IA/NT. Managing this cost is key to **Total Cost of Development** because of the opportunity cost of each dollar spent porting software which could have been spent developing new features in the rapidly growing CFD market.

HOW THE WORK WAS DONE

The application developers had already introduced RISC/UNIX-vendor directives in FLUENT. Based on this work, IA/NT parallel processing was applied to FLUENT with OpenMP directives using KAI's Guide. GuideView allowed the developers of FLUENT to analyze the end user **performance** on Intel SMP systems and compare them to UNIX platforms. KAP/Pro Toolset allowed Fluent developers to shift to IA/NT rapidly and make side-by-side performance comparisons, significantly reducing the **time to market**. The total porting and tuning effort invested by KAI and Fluent Inc. has only been a few weeks.

PERFORMANCE DEMONSTRATION

Data sets from four problems were used in this demonstration:

- STANCAS simulates turbulent flow in a duct elbow.
- UGM2 simulates swirling turbulent flow in a curved duct with a sudden expansion inlet.

- Mixer simulates laminar two-phase flow through a static mixer.
- Burner simulates combustion in a high-speed burner.

The first problem is a standard test case that runs in about 5 minutes on about 40,000 cells. The last three represent typical uses of FLUENT by engineers in practice and run sequentially for 1 to 2 hours on the Intel Architecture workstations running Windows NT using 200K to 300K cells.

The following table illustrates that dual processor Intel Architecture SMP workstations provide a substantial **performance** increase over a single processor workstation on this major engineering analysis application.

	Application			
	STANCAS	UGM2	Mixer	Burner
Pentium® II Xeon™ Processor Scaling	1.81	1.86	1.54	1.76

Table 1 Fluent, FLUENT Scaling from uni-processor to dual-processor

The Intel Pentium II Xeon Processor is a balanced system. It features the fastest clock speed for Intel Architecture processors in addition to supporting a full speed L2 cache interface. Previous Pentium II Processors support L2 cache that runs at half the speed of the processor core. In general, the combination of increased clock speed and faster L2 cache enables multiprocessor workstations based on the Pentium II Xeon Processor to maintain scaling of multi-threaded applications, depending on application-specific thread synchronization and other application-specific factors. Again, application-specific factors will affect overall scaling but this example showcases the Pentium II Xeon Processor as a balanced system capable of delivering improved performance and good scaling.

Oxford Molecular Group, Dgauss*

Oxford Molecular Group (www.oxmol.com) is a leading supplier of software and services for discovery research with computational chemistry, with software installed at all the major chemical and pharmaceutical companies. DGauss is an Oxford Molecular Group application originally developed for Cray supercomputers. Parallel IA/NT workstations now enable experimental chemists to effectively reduce **time to solution** in the development of new chemicals and drugs. The **portability** provided by KAP/Pro Toolset allowed the code to be moved easily from UNIX supercomputers and workstations to Intel Architecture systems.

HOW THE WORK WAS DONE

Guide, part of the KAP/Pro Toolset, enables software developed to run on a Cray to migrate easily to Intel systems running Windows NT. At the same time, with OpenMP a developer can easily increase the number of processors that can be effectively applied to parallel computations. GuideView was used to find and **tune performance** bottlenecks (see footnotes at end: ref. 1, part II). Oxford Molecular Group used Assure, a tool for **debugging** and **Q/A** to bring new functionality to market more quickly on Intel Architecture systems than on supercomputers.

PERFORMANCE DEMONSTRATION

Four molecules in the silicalite series – silicon-containing zeolite subclusters of varying sizes – were modeled. The sequential computation times for these calculations of increasing complexity, were about 1 min., 10 min., 1 hr., and 4 hrs., respectively.

The following table demonstrates the KAP/Pro OpenMP IA/NT parallel workstation **performance** results.

	Silicalite Series Models			
	90 min.	3 hrs.	6 hrs.	12 hrs
Pentium® II Xeon™ Processor Scaling	1.57	1.56	1.58	1.53

Table 2 Oxford Molecular, Dgauss from uni-processor to dual-processor

The scaling on this application is lower than that observed on other applications. This is mainly due to the amount of exploitable parallelism in the application and the effects of Amdahl's Law. The overall speed up is still quite respectable and compares well with previous results observed on the Pentium II processor based workstations.

NCAR, MM5

The National Center for Atmospheric Research (NCAR, www.mm5.ucar.edu/mm5) and Pennsylvania State University have developed MM5 as a weather forecasting model. MM5 is in use by 380 users at over 115 sites worldwide and is distributed by NCAR. This mesoscale modeling program is used for regional weather forecasts in the U.S. and foreign countries, for airport weather forecasting, and by other government agencies. Originally a supercomputer application, MM5 has now been parallelized for Intel Architecture systems running Windows NT.

HOW THE WORK WAS DONE

KAP/Pro Toolset was used to parallelize MM5 by selectively translating parts of two proprietary OEM directive sets that had been introduced. GuideView was used to locate **performance problems** and as a result, some loops were set to run in parallel only under certain runtime data dependent conditions. Assure found a **parallel bug** that had existed in the proprietary directives concerning unsynchronized reductions; this had caused numerical problems that were hard to find in normal debugging of the code.

PERFORMANCE DEMONSTRATION

Four large Sesame Squall-line data sets were used, simulating weather for 90 min., 3 hrs. 6 hrs., and 12 hrs., respectively. The speedup was constant across all four data sets, as shown in the table below. The running time for these applications is several hours. This represents a classical case where parallelism can be used to reduce the computation on IA/NT systems from a duration that nearly equals the simulated time, and is therefore worthless, to nearly half that time, and is therefore useful as a practical forecast. This is an example in which parallelism becomes the ownership enabler in a **TBO** analysis.

	Duration of Weather Simulation			
	90 min.	3 hrs.	6 hrs.	12 hrs
Pentium® II Xeon™ Processor Scaling	1.81	1.82	1.82	1.82

Table 3 NCAR, MM4 Scaling from uni-processor to dual-processor

As before, the scaling of the workstation based on the Pentium II Xeon Processor reflects well on the balance of the system. Like the prior example based on Dgauss, the overall performance of the 400MHz Pentium II Xeon system is faster than the Pentium II systems.

SUMMARY

Intel Architecture multiprocessor workstations and servers with two or more processors are becoming commonplace and demand for multithreaded applications to run on them is growing. This paper shows that porting to Intel Architecture systems from UNIX systems is quite easy. Moreover, the performance of workstations based on the Intel Pentium II Xeon processor rivals many RISC processors. This makes Intel Architecture systems more attractive not only for application delivery but also for initial application development. While there will be application-to-application variances, the scalability of systems from 1 to 2 processors on Intel Architecture systems running Windows NT is comparable to systems based on other architectures.

Two events of the past year offer great opportunity and benefit to application developers using Intel Architecture systems. First, there was the introduction of KAP/Pro Toolset for Intel Architecture workstations running Windows NT, which allows developers the same facilities on IA/NT systems that had been available only on RISC/UNIX systems. Second, the OpenMP standard for parallel programming was introduced. OpenMP has been widely endorsed and adopted by computer manufacturers (including manufacturers of Intel Architecture systems), system software developers, and applications developers. KAP/Pro Toolset for NT provides a complete OpenMP implementation for Fortran developers now and will offer OpenMP for the C/C++ standard as it appears in mid-1998.

The benefit/cost economies of parallel processing on two processor Intel Architecture workstations have now become quite favorable. For properly parallelized applications, one can expect substantial performance gains in return for the low cost of an additional processor and some memory. We have observed that the performance is also quite stable over a range of applications and sizes of problems, as reflected in the examples of this paper. The scalability potential will grow as multi-processor systems become more widely available and as successive generations of Intel Architecture workstations and servers become available. For more information about such Intel Architecture technologies as Pentium II Xeon processors and Merced™, the first in a family of Intel Architecture 64-bit systems (IA-64™), please see the Intel web page: www.Intel.com.

The results reported in this paper should be regarded as a snapshot of work in progress and should be considered as examples. As mentioned before, performance results will vary from application to application, and from data set to data set. Every application in wide use is under continual development on two fronts: more meaningful simulations are being developed and faster software implementations are being introduced. Especially for parallel processing, these two development threads are sometimes in short-term conflict. However, the use of powerful parallel software engineering techniques on widely used, increasingly powerful Intel Architecture systems will lead to a broader understanding and realization of increased benefit and reduced costs in delivering parallel application software.

APPENDIX A — TEST CONFIGURATIONS

Specification	Intel® Pentium® II Xeon™ Processor
Processor	Pentium II Xeon with 512 KB level 2 cache, 400 MHz clock rate, 100 MHz system bus, ECC on
Mother board	Marlinespike* with Alpha 5 BIOS and Intel 82440BX AGPset
Memory	SDRAM, ECC on
Operating System Configuration	Windows* NT* 4.0 1381 SP3 with NTFS file system on disk

Table 4 Basic configuration used for benchmarking the Intel Pentium II Xeon Processor

Benchmark	Memory	Graphics	Disk Subsystem	Networking Card
Dgauss Fluent NCAR MM5	512 MB	Matrox Millennium II, 4MB, 3.11, 4.0.42 Driver, Colors=16	Seagate Cheetah ST34501W, Adaptec 2940 UW Controller	EtherExpress Pro/100B

Table 5 Specific configurations used for the Intel Pentium II Xeon Processor measurements

WEBSITE REFERENCES

- **Intel**
  http://www.intel.com
- **Kuck and Associates, Inc.**
  www.kai.com
- **Fluent**
  www.fluent.com
- **Oxford Molecular Group, Ltd.**
  www.oxmol.com
- **MM5**
  www.mm5.ucar.edu/mm5